Jaden Dawdy
Professor Anderson
CS 457, Intro to AI
October 11, 2024

Grocery Bagging Report

Development Process

At the beginning, this project didn't seem very difficult. In class, there was a lot of informational code given that was very helpful to start the development process. At first, the priority was to just get a working class that properly generated success or failure. Speed was second priority over success. From what was already given, it wasn't very hard to get working code that can determine if something was or wasn't able to fit in the bags.

Once it was able to do small and simple problems correctly, the next step was to increase efficiency. This was a major struggle and after many hours spent, efficiency barely increased.

The first action taken to start increasing efficiency was doing research. Going back to Chapter 6 reading was very helpful in getting started. It gave a lot of good descriptions of constraint satisfaction. There was a lot of struggle to understand how to incorporate it to the problem given.

First objective was to make sure that all items were sorted properly. This improved times significantly especially when paired with backtracking as it took care of a lot of the constrained items beforehand.

Next was to get the backtracking working. The search function was originally using nested loops, and used a brute force algorithm to find the solution to a given problem. To implement backtracking, the search function was switched to use recursion by calling a method called backtrack which is recursive. This would then call a pack method, to see if there were any slots available for a certain item. If it was successful then it would continue to the next item. If it was unsuccessful, it would backtrack and remove the previous item from the bag to try a new orientation. In a way it acts like a more refined version of depth first search.

Next was to implement a form of forward tracking. This was done inside the pack method. A list was created called compatibleBags to help speed up the search process. Inside the pack and unpack method, if an item was successfully inputted or outputted from a bag, it would update compatibleBags for all other items. This is forward tracking because it would prune the search early so that it didn't waste time searching to put items in bags that they couldn't go in.

There were also some error checks that got added. This was brought to attention because of the g7.txt example. There was an issue where the program would run indefinitely trying to find a solution to g7.txt even though it was clear it was a failure. This was because the sum of the weights of the items was greater than the total amount

of weight that can be carried (bag capacity * bag count). This problem was quickly solved by adding an if statement at the beginning of search to determine if there was enough room to fit all items in the first place.

Next the node consistency and arc consistency were implemented based on the book. Node consistency ensures that each item's domain (possible bags it can be packed into) only contains bags that can actually pack the item. It iterates through all the items and for each item, it removes the bags from its domain that can't pack it. If an item's domain becomes empty afterwards, then it means there's nowhere for the item to do so and it returns false indicating that it failed. If all items have at least one bag in their domain it returns true. This helps prevent wasting time looking for what to do when it's not possible in the first place. This is only at the beginning of the search.

The arc consistency check creates a queue of all possible pairs of items. For each arc, it calls the revise method which checks if the domain of the first item needs to be updated based on the constraints with the second item. If *revise* returns true, it checks if the domain of the first item became empty, and if it is, it returns false. If then adds new arcs to the queue for all other items with the first item to propagate the changes. It repeats until the queue is completely empty.

Lastly, two more methods were implemented to increase efficiency. selectUnassignedItems and orderDomainValues. selectUnassignedItems is used to determine what item to choose next to be assigned to a bag. This is very important in backtracking as having an efficient way to decide can allow for faster programs. It first finds the item with the smallest domain size (MRV), if there are items with the same domain size, it picks the one with the most conflicts. If it's still a tie, it picks the one with the most allowed items. For the final tiebreaker, it picks the largest item left.

orderDomainValues helps determine what order to assign bags to items in backtracking. It mainly prefers bags with the most items in them, but if there is a tie, it then picks the bag with the least amount of room left in it.

Testing:

The program was mainly tested using the example test files that were provided in class. There were some tests manually inputted, but it was much more efficient to test using the given ones. The CheckBag.jar was used to determine the accuracy of the program against the files given.

Results

Input from g2.txt:

4
10
item0 6
item1 6
item2 6

item5 3
item6 3
item7 3
item8 2
item9 2
item10 2
item11 2
item12 2
item13 2
item14 1

Output:
success
item0 item14 item5
item2 item13 item10
item1 item8 item12
item7 item9 item11 item6

It took 0.005921 seconds to get the answer and was able to get a correct solution.

Input from g3.txt:
4
10
item0 2 - item1 item4 item2 item5 item3
item1 4 - item4 item3 item0 item5 item2
item2 6 - item3 item5 item0 item4 item1
item3 5 - item5 item2
item4 4 - item2
item5 5

Output:
Failure
It took 0.007275 seconds to get the answer and was correct in its conclusion

Input from g4.txt:
10
10
item0 5 - item1 item2 item3 item4 item5 item6 item7 item8 item9
item1 5 - item0 item2 item3 item4 item5 item6 item7 item8 item9
item2 5 - item0 item1 item3 item4 item5 item6 item7 item8 item9
item3 5 - item0 item1 item2 item4 item5 item6 item7 item8 item9
item4 5 - item0 item1 item2 item3 item5 item6 item7 item8 item9
item5 5 - item0 item1 item2 item3 item4 item6 item7 item8 item9
item6 5 - item0 item1 item2 item3 item4 item5 item7 item8 item9
item7 5 - item0 item1 item2 item3 item4 item5 item6 item8 item9
item8 5 - item0 item1 item2 item3 item4 item5 item6 item7 item9

item9 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8
item10 3
item11 3
item12 3
item13 3
item14 3
item15 3
item16 3
item17 3
item18 3
item19 3

Output:
success
item0 item19
item2 item18
item1 item17
item16 item8
item15 item7
item14 item9
item13 item4
item12 item3
item11 item6
item10 item5

It took 0.007277 seconds to get the correct result.

Input from g5.txt:
20
10
item0 5 - item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item1 5 - item0 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item2 5 - item0 item1 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item3 5 - item0 item1 item2 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item4 5 - item0 item1 item2 item3 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item5 5 - item0 item1 item2 item3 item4 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item6 5 - item0 item1 item2 item3 item4 item5 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item7 5 - item0 item1 item2 item3 item4 item5 item6 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19

item8 5 - item0 item1 item2 item3 item4 item5 item6 item7 item9 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item9 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item10 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item10 item11 item12 item13 item14 item15 item16 item17 item18 item19
item11 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item11 item12 item13 item14 item15 item16 item17 item18 item19
item12 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item12 item13 item14 item15 item16 item17 item18 item19
item13 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item13 item14 item15 item16 item17 item18 item19
item14 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item14 item15 item16 item17 item18 item19
item15 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item15 item16 item17 item18 item19
item16 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item16 item17 item18 item19
item17 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item17 item18 item19
item18 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item18 item19
item19 5 - item0 item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11 item12 item13 item14 item15 item16 item17 item19
item20 3
item21 3
item22 3
item23 3
item24 3
item25 3
item26 3
item27 3
item28 3
item29 3
item30 3
item31 3
item32 3
item33 3
item34 3
item35 3
item36 3
item37 3
item38 3
item39 3

Output:

success
item0 item39
item2 item38
item1 item37
item19 item36
item18 item35
item17 item34
item16 item33
item15 item32
item14 item31
item13 item30
item29 item12
item28 item11
item27 item10
item8 item26
item7 item25
item24 item9
item23 item4
item22 item3
item21 item6
item20 item5

It took 0.011523 seconds to get the correct result.

Analysis:

The program developed for this project did pretty well and was very accurate. There were no times in testing once complete where it gave incorrect results. Although the accuracy was high, it struggled to be very fast. There were several tests that were used that the program just couldn't seem to completely analyze in a reasonable amount of time. The main one of struggle was g1.txt. Several hours were put into finding a way for the program to quickly determine that the output was a failure. Even after reading the whole chapter and doing online research, no solution could be found to get the input to quickly return a correct result.

Experience:

This project was a significant struggle. Up until the day of the project due date, it was planned to be done with a partner, and I worked on the project prior, but the partner didn't want to work until closer to the due date. This led me to wait longer than I felt comfortable. When the due date came, I found out the person who I planned to work with decided not to be my partner anymore so I was left to do the entire project all on my own. That would have been fine if I knew earlier but I worked with the plan of sharing work with another. Although this was frustrating, I did the best I could with the time I had. Overall the project was really interesting and I genuinely enjoyed learning

and getting hands on experience with the search algorithms learned in class. Many attempts were made to increase efficiency and even implement the extra credit but sadly there wasn’t enough time to get what was wanted done.